전공: 컴퓨터공학과 학년: 4 학번: 20212020 이름: 박민준

# 목적

유닉스 시스템에 대하여 미리 접해본 후 실험에 임할 수 있도록 한다. 아울러 부록에 나 와 있는 명령어에 대하여 익숙해지도록 사용해본다.

# 예비 학습

UNIX 시스템에 접속해본 뒤 자신의 홈 디렉토리를 확인해본다.

홈 디렉토리 : /sogang/under/cse20212020

쉘 프로그래밍 실험에서 사용할 데이터 파일인 전화번호부를 만들어본다. 단 데이터 파일의 형식은 실험에서 나온 예제에 따르도록 한다. 5명 이상이 들어가 있는 데이터를 만들 되 vi 에디터를 이용하여 작성한다. 단 파일명은 data로 한다.

박민준|서울시 마포구 신수동 서강대학교 PA관 301호|02-705-3333

손흥민|경기도 수원시 권선구 283-58 2403호|010-2844-4293

BTS|서울시 강남구 봉은사로71길 48|010-7492-3286

봉준호|경기도 용인시 처인구 금령로99번길 1층|010-2138-3871

엄준식|서울시 마포구 신수동 대흥로11안길 20|02-283-8712

위의 예제를 편집하는데 사용한 vi 명령어들을 나열하고, 해당 명령 수행하는 결과를 적 어보도록 한다.

‘mkdir hw1’ 명령어를 통해 1주차 실습자료를 위한 디렉토리를 생성한다.

‘cd hw1’ 명령어를 통해 hw1 디렉토리로 이동한다.

‘vi data’ 명령어를 통해 데이터 파일을 생성한 후 vi 편집기를 실행한다.

‘i’를 입력하여 입력 모드로 전환한다.

5명의 전화번호부 데이터들을 입력한다.

‘Esc’ key를 눌러 입력 모드를 종료한 후 ‘:’을 입력하여 콜론모드로 전환한다.

‘wq’를 입력하여 데이터들을 저장한 후 vi 편집기를 종료한다.

위에서 작성한 데이터 파일을 $home/.data 파일로 복사한다. 복사하기 위하여 사용한 명령들을 적어보도록 한다.

cp data ../.data

$home/.data 파일을 그룹 및 다른 사용자가 아무 권한도 갖지 않도록 권한 변경을 해본 다. 사용한 명령을 적어보도록 한다.

chmod 700 .data

또는

chmod go-rwx .data

디렉토리에 대한 읽기, 쓰기, 실행 권한을 설정해보고 각각이 갖는 의미를 살펴본다.

‘chmod’ 명령어를 통해 디렉토리에 대한 권한 설정을 변경할 수 있다.

파일의 권한에 따라 읽기(read), 쓰기(write), 실행(execute)이 가능하다.

권한 그룹에는 user, group, other 총 3개가 있다.

user는 파일 소유자, group은 소유자 그룹, other은 그 외 기타 사용자를 의미한다.

권한 설정에는 기호 모드와 숫자 모드가 있다.

1. 기호 모드

u, g, o, a 영문자는 파일의 권한 그룹을 의미한다. 이 때, a는 u, g, o 전체를 의미한다.

+, - 기호를 통해 권한을 부여하거나 회수할 수 있다.

예를 들어, data 파일에 대해 group과 other에게 쓰기 권한을 부여하고 싶으면

‘chmod go+w data’ 명령어를 사용하면 된다.

2. 숫자 모드

숫자 모드는 지정된 숫자를 활용해 파일에 권한을 부여하거나 회수한다.

숫자는 read(4), write(2), execute(1)을 사용하며 부여하고 싶은 모든 권한의 값을 합하여 사용한다.

예를 들어, file2 파일에 대해 user, group, other에게 실행 권한을 부여하고 싶으면

‘chmod 111 file2’ 명령어를 사용하면 된다.

# 보충 학습

Regular Expression에 대하여 정리해보도록 한다.

정규 표현식이란 특정 규칙을 가진 문자열 집합을 표현하기 위한 형식 언어로서, 주로 데이터 검색과 복잡한 문자열 패턴 매칭 검사 등을 도와주는 특별한 문자이다.

문자열 검색에 정규 표현식을 적용하게 되면 ‘지정된 문자열의 문자가 단순히 같은지’ 여부가 검사되는 것이 아니라, ‘정규 표현식의 규칙에 매칭되는지’ 여부가 검사된다.

정규 표현식(Regular Expression)은 ‘regexp’ or ‘regex’ 로 줄여서 표현하기도 한다.

정규 표현식에는 기본적으로 3가지 유형이 존재한다.

간단하게 정리해보면 다음과 같다.

1. 기본 정규 표현식(Basic Regular Expressions)

. : 점의 개수만큼 아무 문자나 대체 가능한다.

^ : 문자열의 처음 시작 부분을 매칭한다.

$ : 문자열의 끝 부분을 매칭한다.

\* : \* 앞의 문자와 매칭한다.

\ : 특수 문자와 매칭한다.

( ) : 정규 표현식 그룹을 의미한다.

? : 정확히 한 개의 문자와 매칭한다.

2. 간격 정규 표현식(Interval Regular Expressions)

{n} : 앞의 문자와 ‘n’번 매칭한다.

{n, m} : 앞의 문자와 ‘n’번 매칭하되, ‘m’번 이하로 매칭한다.

{n, } : 앞의 문자와 ‘n’번 이상 매칭한다.

3. 확장 정규 표현식(Extended Regular Expressions)

\+ : \+ 앞의 문자가 한 번 이상 출현한 문자열과 매칭한다.

\? : \? 앞의 문자가 1번 이하로 출현한 문자열과 매칭한다.

실무에서는 굳이 정규 표현식을 사용하지 않고 여러 명령어를 통해 원하는 결과를 얻을 수 있다. 또한 매우 간결하게 표현하는 만큼 가독성이 떨어져 잘 모르는 사람이 이해하기 어려울 수 있다.

그러나, 정규 표현식을 사용하게 되면 보다 쉽고 간단하게 원하는 결과를 얻을 수 있기 때문에 리눅스뿐만 아니라 다양한 언어에서 사용되고 있다.